

```

NNN      NNN      EEEEEEEEEEEEEEE  TTTTTTTTTTTTTTT  AAAAAAAAAAA  CCCCCCCCCCCC  PPPPPPPPPPPP
NNN      NNN      EEEEEEEEEEEEEEE  TTTTTTTTTTTTTTT  AAAAAAAAAAA  CCCCCCCCCCCC  PPPPPPPPPPPP
NNN      NNN      EEEEEEEEEEEEEEE  TTTTTTTTTTTTTTT  AAAAAAAAAAA  CCCCCCCCCCCC  PPPPPPPPPPPP
NNN      NNN      EEE              TTT              AAA              AAA  CCC              PPP              PPP
NNN      NNN      EEE              TTT              AAA              AAA  CCC              PPP              PPP
NNN      NNN      EEE              TTT              AAA              AAA  CCC              PPP              PPP
NNNNNN   NNN      EEE              TTT              AAA              AAA  CCC              PPP              PPP
NNNNNN   NNN      EEE              TTT              AAA              AAA  CCC              PPP              PPP
NNNNNN   NNN      EEE              TTT              AAA              AAA  CCC              PPP              PPP
NNN      NNN      EEEEEEEEEEEEEEE  TTT              AAA              AAA  CCC              PPPPPPPPPPPPP
NNN      NNN      EEEEEEEEEEEEEEE  TTT              AAA              AAA  CCC              PPPPPPPPPPPPP
NNN      NNN      EEEEEEEEEEEEEEE  TTT              AAA              AAA  CCC              PPPPPPPPPPPPP
NNN      NNN      EEE              TTT              AAA              AAA  CCC              PPP
NNN      NNN      EEE              TTT              AAA              AAA  CCC              PPP
NNN      NNN      EEE              TTT              AAA              AAA  CCC              PPP
NNN      NNN      EEE              TTT              AAA              AAA  CCC              PPP
NNN      NNN      EEE              TTT              AAA              AAA  CCC              PPP
NNN      NNN      EEEEEEEEEEEEEEE  TTT              AAA              AAA  CCCCCCCCCCCC  PPP
NNN      NNN      EEEEEEEEEEEEEEE  TTT              AAA              AAA  CCCCCCCCCCCC  PPP
NNN      NNN      EEEEEEEEEEEEEEE  TTT              AAA              AAA  CCCCCCCCCCCC  PPP

```

NE  
VC

```

LL                      IIIIII                      SSSSSSSSS
LL                      IIIIII                      SSSSSSSSS
LL                      II                      SS
LL                      II                      SS
LL                      II                      SS
LL                      II                      SS
LL                      II                      SSSSSSS
LL                      II                      SSSSSSS
LL                      II                      SS
LL                      II                      SS
LL                      II                      SS
LL                      II                      SS
LLLLLLLLLLLLLLLL      IIIIII                      SSSSSSSSS
LLLLLLLLLLLLLLLL      IIIIII                      SSSSSSSSS

```



```

1 0001 0 MODULE network_server (IDENT = 'V04-000',
2 0002 0 MAIN = network_server,
3 0003 0 ADDRESSING_MODE(EXTERNAL=GENERAL)) =
4 0004 1 BEGIN
5 0005 1
6 0006 1
7 0007 1
8 0008 1
9 0009 1
10 0010 1
11 0011 1
12 0012 1
13 0013 1
14 0014 1
15 0015 1
16 0016 1
17 0017 1
18 0018 1
19 0019 1
20 0020 1
21 0021 1
22 0022 1
23 0023 1
24 0024 1
25 0025 1
26 0026 1
27 0027 1
28 0028 1
29 0029 1
30 0030 1
31 0031 1
32 0032 1
33 0033 1
34 0034 1
35 0035 1
36 0036 1
37 0037 1
38 0038 1
39 0039 1
40 0040 1
41 0041 1
42 0042 1
43 0043 1
44 0044 1
45 0045 1
46 0046 1
47 0047 1
48 0048 1
49 0049 1
50 0050 1
51 0051 1
52 0052 1
53 0053 1
54 0054 1
55 0055 1
56 0056 1
57 0057 1

*****
*
* COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
* DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
* ALL RIGHTS RESERVED.
*
* THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
* ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
* INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
* COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
* OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
* TRANSFERRED.
*
* THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
* AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
* CORPORATION.
*
* DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
* SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
*
*****

++
FACILITY: DECnet

ABSTRACT:

This program is used to enable a process to wait for an incoming
DECnet logical link connection, and then accept the logical link
request by invoking the correct procedure using CLI CHAIN. This
is used to allow a single process to handle many logical link
requests, and reduce the overhead involved in process creation.

ENVIRONMENT:

VAX/VMS operating system. unprivileged user mode,

AUTHOR: Tim Halvorsen, June 1982

Modified by:

V03-004 PRB0337 Paul Beck 27-Jun-1984 16:33
Change default timeout from 1 minute to 5 minutes.

V003 TMH0003 Tim Halvorsen 07-Apr-1983
Add support for direct execution of an object image,
if the object filespec contains an explicit ".EXE".

V002 TMH0002 Tim Halvorsen 24-Feb-1983
Add support for EPIDs by using the IPID returned

```

NETWORK\_SERVER  
V04-000

H 5  
16-Sep-1984 01:39:23  
14-Sep-1984 12:49:31

VAX-11 Bliss-32 V4.0-742  
[NETACP.SRC]SERVER.B32;1

Page 2  
(1)

```
: 58      0058 1 | by DECLSERV to index the SPI database, rather than
: 59      0059 1 | using the EPID returned by GETJPI.
: 60      0060 1 |
: 61      0061 1 | V001 TMH0001 Tim Halvorsen 7-Feb-1983
: 62      0062 1 | Add code to display where each connect request comes
: 63      0063 1 | from (by displaying the NCB), so that .LOG files can
: 64      0064 1 | be more easily read.
: 65      0065 1 | --
: 66      0066 1 |
: 67      0067 1 |
: 68      0068 1 | Include files
: 69      0069 1 |
: 70      0070 1 |
: 71      0071 1 | LIBRARY 'SYSS$LIBRARY:STARLET';      ! VAX/VMS common definitions
: 72      0072 1 |
: 73      0073 1 | LIBRARY 'SHRLIB$:NET';              ! NETACP control QIO definitions
```



```

75      0074 1  |
76      0075 1  | Table of contents
77      0076 1  |
78      0077 1  |
79      0078 1  | FORWARD ROUTINE
80      0079 1  |   network_server,
81      0080 1  |   timeout_ast: NOVALUE,
82      0081 1  |   issue_mailbox_read: NOVALUE,
83      0082 1  |   net_interrupt: NOVALUE,
84      0083 1  |   fao_buffer;
85      0084 1  |
86      0085 1  |
87      0086 1  | Literals
88      0087 1  |
89      0088 1  |
90      0089 1  | LITERAL
91      0090 1  |   true = 1;
92      0091 1  |   false = 0;
93      0092 1  |
94      0093 1  |
95      0094 1  | Macros
96      0095 1  |
97      0096 1  |
98      0097 1  | MACRO
99      M 0098 1  |   fao(string) =
100     M 0099 1  |   fao_buffer(%ASCID string
101     0100 1  |   %IF %LENGTH GTR 1 %THEN ,%REMAINING %FI)%,
102     0101 1  |
103     M 0102 1  |   write_line(string) =
104     M 0103 1  |   LIB$PUT_OUTPUT(fao(string
105     0104 1  |   %IF %LENGTH GTR 1 %THEN ,%REMAINING %FI))%,
106     0105 1  |
107     M 0106 1  |   signal_if_error(command) =
108     M 0107 1  |   BEGIN
109     M 0108 1  |   LOCAL
110     M 0109 1  |   status;
111     M 0110 1  |
112     M 0111 1  |   status = command;
113     M 0112 1  |   IF NOT .status
114     M 0113 1  |   THEN
115     M 0114 1  |   BEGIN
116     M 0115 1  |   SIGNAL(.status);
117     M 0116 1  |   RETURN .status OR sts$m_inhib_msg;
118     M 0117 1  |   END;
119     0118 1  |   END%;
120     0119 1  |
121     0120 1  |
122     0121 1  | Own storage
123     0122 1  |
124     0123 1  |
125     0124 1  | LITERAL
126     0125 1  |   mbx_maxmsg = 128;
127     0126 1  |
128     0127 1  | OWN
129     0128 1  |   net_channel: WORD,
130     0129 1  |   mbx_channel: WORD,
131     0130 1  |   mbx_message: VECTOR [mbx_maxmsg,BYTE],

```

```

| Main routine
| Timeout AST
| Issue network mailbox read
| Mailbox attention AST
| Invoke FAO and return descriptor

```

! Maximum size of mailbox message

```

| Channel to ACP
| Channel to assoc. mailbox
| Mailbox input buffer

```

NETWORK\_SERVER  
V04-000

J 5  
16-Sep-1984 01:39:23 VAX-11 Bliss-32 V4.0-742  
14-Sep-1984 12:49:31 [NETACP.SRC]SERVER.B32;1

Page 4  
(2)

```
: 132      0131 1      mbx_iosb:  $BLOCK [8];
: 133      0132 1
: 134      0133 1
: 135      0134 1      External routines
: 136      0135 1
: 137      0136 1
: 138      0137 1      EXTERNAL ROUTINE
: 139      0138 1      lib$asn_wth_mbx,
: 140      0139 1      lib$set_logical,
: 141      0140 1      lib$run_program,
: 142      0141 1      lib$do_command,
: 143      0142 1      lib$put_output,
: 144      0143 1      str$concat;
```

! I/O status block for mailbox

```
! Assign with assoc. mailbox
! Define supervisor mode logical name
! Chain to another program
! Chain a CLI command string
! Write to SYS$OUTPUT
! Concatenate strings together
```



```
146 0144 1 ROUTINE network_server =
147 0145 1
148 0146 1 |---
149 0147 1 |
150 0148 1 |       This routine is the entry point to the program
151 0149 1 |
152 0150 1 |       Inputs:
153 0151 1 |
154 0152 1 |       None
155 0153 1 |
156 0154 1 |       Outputs:
157 0155 1 |
158 0156 1 |       Routine value = status code
159 0157 1 |---
160 0158 1
161 0159 2 BEGIN
162 0160 2
163 0161 2 LOCAL
164 0162 2     nfb:          $BBLOCK [nfb$length+20*4],      ! Network function block
165 0163 2                                     ! (room for 20 field requests)
166 0164 2     nfb_desc:    VECTOR [2]                      ! Descriptor of NFB
167 0165 2               INITIAL(nfb$length + 3*4),
168 0166 2     iosb:        $BBLOCK [8],                    ! I/O status block
169 0167 2     time_buf:   VECTOR [128,BYTE],              ! Buffer for timeout specifier
170 0168 2     time_desc:  VECTOR [2]                      ! Descriptor of timeout specifier
171 0169 2               INITIAL(128),
172 0170 2     delta_time: VECTOR [2],                      ! Binary time quadword
173 0171 2     buffer:      VECTOR [64],                    ! Return buffer
174 0172 2     buffer_desc: VECTOR [2]                    ! Descriptor of above buffer
175 0173 2               INITIAL(256),
176 0174 2     keys:        $BBLOCK [4+4+nfb$ctx_size], ! Buffer for search key & context
177 0175 2     key_desc:    VECTOR [2]                      ! Descriptor of above buffer
178 0176 2               INITIAL(4+4+nfb$ctx_size),
179 0177 2     ptr:          REF $BBLOCK,                      ! Pointer into return buffer
180 0178 2     cmd_desc:    $BBLOCK [8]                      ! Command string
181 0179 2               PRESET ([dsc$b_class] = dsc$k_class_d,
182 0180 2                       [dsc$w_length] = 0,
183 0181 2                       [dsc$a_pointer] = 0),
184 0182 2     ncb_desc:     VECTOR [2],                      ! Descriptor of NCB
185 0183 2     ascii_ncb_desc: VECTOR [2],                    ! Descriptor of ASCII portion of NCB
186 0184 2     filespec:     VECTOR [2],                      ! Descriptor of procedure filespec
187 0185 2     prcnam:       VECTOR [2],                      ! Descriptor of process name
188 0186 2     ipid,          ! Our IPID
189 0187 2     epid,          ! Our EPID
190 0188 2     item_list:    $BBLOCK [10*4]
191 0189 2               PRESET ([0,0,16,0] = 4,
192 0190 2                       [2,0,16,0] = jpi$_pid,
193 0191 2                       [8,0,32,0] = 0,
194 0192 2                       [12,0,32,0] = 0),
195 0193 2     status:
196 0194 2
197 0195 2 BIND
198 0196 2     default_time = %ASCID '0 00:05:00': $BBLOCK;
199 0197 2
200 0198 2
201 0199 2 | Initialize some stack local variables with dynamic pointers
202 0200 2 |
```

```

203      0201      2
204      0202      2 nfb_desc [1] = nfb;
205      0203      2 time_desc [1] = time_buf;
206      0204      2 buffer_desc [1] = buffer;
207      0205      2 key_desc [1] = keys;
208      0206      2 item_list [4,0,32,0] = epid;
209      0207      2
210      0208      2
211      0209      2 : Get our own EPID for later lookup of our server parameters
212      0210      2
213      0211      2
214      P 0212      2 signal_if_error(
215      0213      2     $GETJPI(ITMLST = item_list));      ! Get our EPID
216      0214      2
217      0215      2
218      0216      2 : Assign a channel to the network ACP
219      0217      2
220      0218      2
221      P 0219      2 signal_if_error(
222      P 0220      2     LIB$ASN_WTH_MBX(%ASCID '_NET:',      ! Assign channel to NETACP
223      P 0221      2     0,0,      ! mailbox MAXMSG,BUFQUO
224      P 0222      2     net_channel,      ! Channel to NETACP
225      0223      2     mbx_channel));      ! Channel to mailbox
226      0224      2
227      0225      2
228      0226      2 : Issue a read on the associated mailbox, so that we can receive
229      0227      2 : notification of network broadcast messages. This is done so that
230      0228      2 : we can detect the network shutting down.
231      0229      2
232      0230      2
233      0231      2 issue_mailbox_read();      ! Issue mailbox read
234      0232      2
235      0233      2
236      0234      2 : Set our process name to something which indicates that we are a network
237      0235      2 : server waiting for work. This has the effect of wiping out the previous
238      0236      2 : process name set by the previous connect to this process.
239      0237      2
240      0238      2
241      0239      2 prcnam [0] = .buffer_desc [0];      ! Make descriptor of scratch buffer
242      0240      2 prcnam [1] = .buffer_desc [1];
243      0241      2
244      P 0242      2 $FAO(%ASCID 'SERVER!XW',      ! Generate a unique process name
245      P 0243      2     prcnam,      ! Output buffer descriptor
246      P 0244      2     prcnam [0],      ! Place to return length
247      0245      2     .epid);      ! Use last 4 digits of EPID
248      0246      2
249      0247      2 $SETPRN(PRCNAM = prcnam);      ! Set our process name
250      0248      2 : (ignore any errors)
251      0249      2
252      0250      2
253      0251      2 : Schedule a timer, so that if the following QIO does not complete within
254      0252      2 : a reasonable amount of time, we can go away (since there was no work to do).
255      0253      2
256      0254      2
257      P 0255      2 status = $TRNLOG(LOGNAM = %ASCID 'NETSERVER$TIMEOUT', ! Get timeout value
258      P 0256      2     RSLBUF = time_desc,
259      0257      2     RSLLEN = time_desc [0]);

```



```

260 0258 2
261 0259 2 IF .status NEQ ss$_normal ! If not explicitly specified,
262 0260 2 THEN
263 0261 2 BEGIN
264 0262 2     time_desc [0] = .default_time [dsc$_length];
265 0263 2     time_desc [1] = .default_time [dsc$_pointer];
266 0264 2 END;
267 0265 2
268 P 0266 2 signal if_error(
269 P 0267 2     $BINTIM(TIMBUF = time_desc, ! Translate time specifier to binary
270 0268 2     TIMADR = delta_time));
271 0269 2
272 P 0270 2 signal if_error(
273 P 0271 2     $SETIMR(DAYTIM = delta_time, ! Start timer
274 0272 2     ASTADR = timeout_ast)); ! Address of AST routine
275 0273 2
276 0274 2 !
277 0275 2 ! Tell NETACP that we are available for a connect request. The QIOW
278 0276 2 ! will complete when a connect has been assigned to us.
279 0277 2 !
280 0278 2
281 0279 2 CH$FILL(0,nfb$_length,nfb); ! Pre-zero NFB fields
282 0280 2 nfb [nfb$_fct] = nfb$_declserv; ! Tell NETACP we are available for work
283 0281 2
284 P 0282 2 status = $QIOW(FUNC = IOS$_ACPCONTROL, ! Issue control function
285 P 0283 2     CHAN = .net_channel,
286 P 0284 2     IOSB = iosb,
287 0285 2     P1 = nfb_desc); ! Address of NFB descriptor
288 0286 2
289 0287 2 IF NOT .status ! If error detected,
290 0288 2 OR NOT (status = .iosb [0,0,16,0])
291 0289 2 THEN
292 0290 2     IF .status EQL ss$_abort ! If we timed out,
293 0291 2     THEN
294 0292 2         BEGIN
295 0293 2             $DASSGN(CHAN = .net_channel); ! Deassign the ACP channel
296 0294 2             RETURN sts$_severe OR sts$_inhib_msg; ! Return "fatal" from program
297 0295 2         END
298 0296 2     ELSE
299 0297 2         BEGIN
300 0298 2             SIGNAL(.status); ! else signal the error
301 0299 2             $DASSGN(CHAN = .net_channel); ! Deassign the ACP channel
302 0300 2             RETURN true;
303 0301 2         END;
304 0302 2
305 0303 2 ipid = .iosb [4,0,32,0]; ! Get our IPID returned by DECLSERV
306 0304 2
307 0305 2 CH$FILL(0,nfb$_length,nfb); ! Pre-zero NFB fields
308 0306 2
309 0307 2 nfb [nfb$_fct] = nfb$_fc_show; ! Request "show" function
310 0308 2 nfb [nfb$_database] = nfb$_db_spi; ! of server process database
311 0309 2 nfb [nfb$_srch_key] = nfb$_spi_pid; ! for our process
312 0310 2 nfb [nfb$_oper] = nfb$_op_eq; ! by checking if field EQL P2 value
313 0311 2
314 0312 2 CH$MOVE(4*4, UPLIT LONG( ! Request the following fields:
315 0313 2     nfb$_spi_ncb, ! Network connect block
316 0314 2     nfb$_spi_sfi, ! Procedure filespec

```

```

317      0315 2      nfb$sc_spi_pnm,      ! Process name
318      0316 2      nfb$sc_endoflist),
319      0317 2      nfb [nfb$sl_fldid]);
320      0318 2
321      0319 2      keys [0,0,32,0] = 0;      ! Zero count of fields in P4 (unused)
322      0320 2      keys [4,0,32,0] = .ipid;      ! Search value = our IPID
323      0321 2      keys [8,0,16,0] = 0;      ! Context area = at beginning
324      0322 2
325      P 0323 2      status = $QIOW(FUNC = IOS$ACPCONTROL,      ! Issue control function
326      P 0324 2      CHAN = .net_channel,
327      P 0325 2      IOSB = iosb,
328      P 0326 2      P1 = nfb_desc,      ! Address of NDB descriptor
329      P 0327 2      P2 = key_desc,      ! Address of key buffer descriptor
330      0328 2      P4 = buffer_desc);      ! Address of return buffer descriptor
331      0329 2
332      0330 2      IF NOT .status      ! If error detected,
333      0331 3      OR NOT (status = .iosb [0,0,16,0])
334      0332 2      THEN
335      0333 2      BEGIN
336      0334 2      SIGNAL(.status);      ! then stop looping
337      0335 2      $DASSGN(CHAN = .net_channel);      ! Deassign the ACP channel
338      0336 2      RETURN true;
339      0337 2      END;
340      0338 2
341      0339 2      ptr = buffer [0];      ! Point to first string in buffer
342      0340 2
343      0341 2      ncb_desc [0] = .ptr [0,0,16,0];      ! Construct descriptor of NCB
344      0342 2      ncb_desc [1] = .ptr + 2;
345      0343 2      ptr = .ptr + 2 + .ptr [0,0,16,0];      ! Skip by string in buffer
346      0344 2
347      0345 2      filespec [0] = .ptr [0,0,16,0];      ! Construct descriptor of procedure
348      0346 2      filespec [1] = .ptr + 2;
349      0347 2      ptr = .ptr + 2 + .ptr [0,0,16,0];      ! Skip by string in buffer
350      0348 2
351      0349 2      prcnam [0] = .ptr [0,0,16,0];      ! Construct descriptor of process name
352      0350 2      prcnam [1] = .ptr + 2;
353      0351 2      ptr = .ptr + 2 + .ptr [0,0,16,0];      ! Skip by string in buffer
354      0352 2
355      0353 2      ptr = CH$FIND_CH(.ncb_desc [0], .ncb_desc [1], '/');
356      0354 2
357      0355 2      ascii_ncb_desc [0] = .ptr - .ncb_desc [1];
358      0356 2      ascii_ncb_desc [1] = .ncb_desc [1];
359      0357 2
360      0358 2      write_line('');
361      0359 2      write_line('-----');
362      0360 2      write_line('');
363      0361 2      write_line('Connect request received at !XD', 0);
364      0362 2      write_line('from remote process !AS'', ascii_ncb_desc);
365      0363 2      write_line('for object !AS'', filespec);
366      0364 2      write_line('');
367      0365 2      write_line('-----');
368      0366 2      write_line('');
369      0367 2
370      P 0368 2      signal_if_error(
371      0369 2      $SETPRN(PRCNAM = prcnam));      ! Set our process name
372      0370 2
373      P 0371 2      signal_if_error(

```



```
374 P 0372 2 LIB$SET_LOGICAL(%ASCID 'SYS$NET', ! Define SYS$NET to NCB
375 0373 ncb_desc));
376 0374
377 0375 cmd_desc [dsc$b_class] = dsc$b_class_d; ! Create dynamic string descriptor
378 0376 cmd_desc [dsc$a_pointer] = 0; ! Indicate no dynamic string yet
379 P 0377 2 signal if error
380 0378 STR$CONCAT(cmd_desc, ! Create '@filespec' command
381 0379 %ASCID 'a',filespec));
382 0380
383 0381 IF NOT CH$FAIL(CH$FIND_SUB( ! If .EXE found in filespec,
384 0382 filespec [0],filespec [1],
385 0383 4, UPLIT BYTE('.EXE')))
386 0384 THEN
387 P 0385 2 signal if error(
388 0386 LIB$RUN_PROGRAM(filespec)) ! Chain to program (EXIT AND CHAIN)
389 0387 ELSE
390 P 0388 2 signal if error(
391 0389 LIB$DO_COMMAND(cmd_desc)); ! Else, chain to command line
392 0390
393 0391 !
394 0392 Do not put any code after this point. Both LIB$RUN_PROGRAM and
395 0393 LIB$DO_COMMAND do not return, then cause immediately program exit.
396 0394 The only way we get here is if they fail.
397 0395 !
398 0396
399 0397 RETURN true; ! Return successfully
400 0398
401 0399 1 END;
```

INFO#250 L1:0245  
Referenced LOCAL symbol EPID is probably not initialized

										.TITLE	NETWORK_SERVER	
										.IDENT	\V04-000\	
										.PSECT	\$PLITS,NOWRT,NOEXE,2	
										0319 0004 00000 P.AAA:	.WORD 4 793	
										00# 00004	.BYTE 0[4]	
00	00	30	30	3A	35	30	3A	30	30	00000000 00000000 00008	P.AAC:	.LONG 0 0
										010E000A 0001C	P.AAB:	.ASCII \0 00:05:00\<0><0>
										00000000' 00020	P.AAB:	.LONG 17694730
										00 00 00 3A 54 45 4E 5F 00024	P.AAE:	.ADDRESS P.AAC
										010E0005 0002C	P.AAD:	.ASCII \ NET:\<0><0><0>
										00000000' 00030	P.AAD:	.LONG 17694725
00	00	57	58	21	5F	52	45	56	52	00034	P.AAG:	.ADDRESS P.AAE
										010E000A 00040	P.AAG:	.ASCII \SERVER !XW\<0><0>
										00000000' 00044	P.AAF:	.LONG 17694730
4F	45	4D	49	54	24	52	45	56	52	00048	P.AAI:	.ADDRESS P.AAG
										00 00 00 54 55 00057	P.AAI:	.ASCII \NETSERVER\$TIMEOUT\<0><0><0>
										010E0011 0005C	P.AAH:	.LONG 17694737
										00000000' 00060	P.AAH:	.ADDRESS P.AAI
										00000000 12020045 12020043 12020044 00064	P.AAJ:	.LONG 302121028, 302121027, 302121029, 0
										00074	P.AAL:	.BLKB 0
										010E0000 00074	P.AAK:	.LONG 17694720
										00000000' 00078	P.AAK:	.ADDRESS P.AAL



NETWORK\_SERVER  
V04-000

C 6  
16-Sep-1984 01:39:23  
14-Sep-1984 12:49:31

VAX-11 Bliss-32 V4.0-742  
[NETACP.SRC]SERVER.B32;1

Page 10  
(3)

```
2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 0007C P.AAN: .ASCII \ -----\
2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 0008B
2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 0009A
2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 000A4 .ASCII \-----\
2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 000B3
010E0040 000BC P.AAM: .LONG 17694784
00000000 000C0 .ADDRESS P.AAN
010E0030 000C4 P.AAP: .BLKB 0
00000000 000C4 P.AAO: .LONG 17694720
00000000 000C8 .ADDRESS P.AAP
74 63 65 6E 6E 6F 43 20 20 20 20 20 20 20 20 000CC P.AAR: .ASCII \ Connect request received at !%D-
76 69 65 63 65 72 20 74 73 65 75 71 65 72 20 000DB \<0>
00 44 25 21 20 74 61 20 64 65 000EA
010E0027 000F4 P.AAQ: .LONG 17694759
00000000 000F8 .ADDRESS P.AAR
6F 72 66 20 20 20 20 20 20 20 20 20 20 20 20 000FC P.AAT: .ASCII \ from remote process !AS'\
73 65 63 6F 72 70 20 65 74 6F 6D 65 72 20 6D 0010B
22 53 41 21 20 73 0011A
010E0024 00120 P.AAS: .LONG 17694756
00000000 00124 .ADDRESS P.AAT
72 6F 66 20 20 20 20 20 20 20 20 20 20 20 20 00128 P.AAV: .ASCII \ for object '!AS'\
22 53 41 21 22 20 74 63 65 6A 62 6F 20 00137
010E001C 00144 P.AAU: .LONG 17694748
00000000 00148 .ADDRESS P.AAV
010E0000 0014C P.AAX: .BLKB 0
00000000 0014C P.AAW: .LONG 17694720
00000000 00150 .ADDRESS P.AAX
2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 00154 P.AAZ: .ASCII \ -----\
2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 00163
2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 00172
2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 0017C .ASCII \-----\
2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 0018B
010E0040 00194 P.AAY: .LONG 17694784
00000000 00198 .ADDRESS P.AAZ
010E0000 0019C P.ABB: .BLKB 0
00000000 0019C P.ABA: .LONG 17694720
00000000 001A0 .ADDRESS P.ABB
00 54 45 4E 24 53 59 53 001A4 P.ABD: .ASCII \SYSSNET\<0>
010E0007 001AC P.ABC: .LONG 17694727
00000000 001B0 .ADDRESS P.ABD
00 00 00 40 001B4 P.ABF: .ASCII \a\<0><0><0>
010E0001 001B8 P.ABE: .LONG 17694721
00000000 001BC .ADDRESS P.ABF
45 58 45 2E 001C0 P.ABG: .ASCII \.EXE\
```

.PSECT \$OWNS,NOEXE,2

```
00000 NET_CHANNEL:
      .BLKB 2
00002 MBX_CHANNEL:
      .BLKB 2
00004 MBX_MESSAGE:
      .BLKB 128
00084 MBX_IOSB:
      .BLKB 8
```

DEFAULT\_TIME= P.AAB



```
.EXTRN LIB$ASN_WTH_MBX
.EXTRN LIB$SET-LOGICAL
.EXTRN LIB$RUN-PROGRAM
.EXTRN LIB$DO_COMMAND, LIB$PUT_OUTPUT
.EXTRN STR$CONCAT, SY$$GETJPI
.EXTRN SY$$FAO, SY$$SETPRN
.EXTRN SY$$TRNLOG, SY$$BINTIM
.EXTRN SY$$SETIMR, SY$$QIOW
.EXTRN SY$$DASSGN
```

.PSECT \$CODE\$,NOWRT,2

OFFC 00000 NETWORK\_SERVER:

	5B	0000'	CF	9E	00002	.WORD	Save R2,R3,R4,R5,R6,R7,R8,R9,R10,R11	:	0144
	5A	0000V	CF	9E	00007	MOVAB	NET_CHANNEL, R11	:	
	59	00000000G	00	9E	0000C	MOVAB	FAO_BUFFER, R10	:	
	58	0000'	CF	9E	00013	MOVAB	LIB\$PUT_OUTPUT, R9	:	
	5E	FD54	CE	9E	00018	MOVAB	P.AAA, R8	:	
98	AD		1C	7D	0001D	MOVQ	-684(SP), SP	:	
FF08	CD	80	8F	9A	00021	MOVZBL	#28, NFB_DESC	:	0159
		FF0C	CD	D4	00027	CLRL	#128, TIME_DESC	:	
00A4	CE	0100	8F	3C	0002B	MOVZWL	TIME_DESC+4	:	
		00A8	CE	D4	00032	CLRL	#256, BUFFER_DESC	:	
54	AE	48	8F	9A	00036	MOVZBL	BUFFER_DESC+4	:	
		58	AE	D4	0003B	CLRL	#72, KEY_DESC	:	
4C	AE	02000000	8F	D0	0003E	MOVL	KEY_DESC+4	:	
		50	AE	D4	00046	CLRL	#33554432, CMD_DESC	:	0181
	68		10	2C	00049	MOVCS	CMD_DESC+4	:	
		04	AE		0004E		#16, P.AAA, #0, #40, ITEM_LIST	:	0192
9C	AD	A0	AD	9E	00050	MOVAB	NFB, NFB_DESC+4	:	0202
FF0C	CD	FF10	CD	9E	00055	MOVAB	TIME_BUF, TIME_DESC+4	:	0203
00A8	CE	00AC	CE	9E	0005C	MOVAB	BUFFER, BUFFER_DESC+4	:	0204
58	AE	5C	AE	9E	00063	MOVAB	KEYS, KEY_DESC+4	:	0205
08	AE		6E	9E	00068	MOVAB	EPID, ITEM_LIST+4	:	0206
			7E	7C	0006C	CLRQ	-(SP)	:	0213
			7E	D4	0006E	CLRL	-(SP)	:	
		10	AE	9F	00070	PUSHAB	ITEM_LIST	:	
			7E	7C	00073	CLRQ	-(SP)	:	
			7E	D4	00075	CLRL	-(SP)	:	
00000000G	00		07	FB	00077	CALLS	#7, SY\$\$GETJPI	:	
	52		50	D0	0007E	MOVL	R0, STATUS	:	
	7A		52	E9	00081	BLBC	STATUS, 2\$	:	
		02	AB	9F	00084	PUSHAB	MBX_CHANNEL	:	0223
			5B	DD	00087	PUSHL	R11	:	
			7E	7C	00089	CLRQ	-(SP)	:	
		2C	A8	9F	0008B	PUSHAB	P.AAD	:	
00000000G	00		05	FB	0008E	CALLS	#5, LIB\$ASN_WTH_MBX	:	
	52		50	D0	00095	MOVL	R0, STATUS	:	
	7C		52	E9	00098	BLBC	STATUS, 3\$	:	
0000V	CF		00	FB	0009B	CALLS	#0, ISSUE_MAILBOX_READ	:	0231
2C	AE	00A4	CE	7D	000A0	MOVQ	BUFFER_DESC, PRCNAM	:	0239
			6E	DD	000A6	PUSHL	EPID	:	0245
		30	AE	9F	000A8	PUSHAB	PRCNAM	:	
		34	AE	9F	000AB	PUSHAB	PRCNAM	:	
		40	A8	9F	000AE	PUSHAB	P.AAF	:	
00000000G	00		04	FB	000B1	CALLS	#4, SY\$\$FAO	:	



	00000000G	00	2C	AE 9F 000B8	PUSHAB	PRCNAM	: 0247
				01 FB 000BB	CALLS	#1, SYS\$SETPRN	: 0257
				7E 7C 000C2	CLRQ	-(SP)	: 0259
				7E D4 000C4	CLRL	-(SP)	: 0262
			FF08	CD 9F 000C6	PUSHAB	TIME_DESC	: 0263
			FF08	CD 9F 000CA	PUSHAB	TIME_DESC	: 0268
			5C	A8 9F 000CE	PUSHAB	P.AAR	: 0272
	00000000G	00		06 FB 000D1	CALLS	#6, SYS\$TRNLOG	: 0279
		56		50 D0 000D8	MOVL	R0, STATUS	: 0280
		01		56 D1 000DB	CMPL	STATUS, #1	: 0285
				0C 13 000DE	BEQL	1\$	: 0287
	FF08	CD	1C	A8 3C 000E0	MOVZWL	DEFAULT_TIME, TIME_DESC	: 0288
	FF0C	CD	20	A8 D0 000E6	MOVL	DEFAULT_TIME+4, TIME_DESC+4	: 0289
			FF00	CD 9F 000EC	PUSHAB	DELTA_TIME	: 0290
			FF08	CD 9F 000F0	PUSHAB	TIME_DESC	: 0293
	00000000G	00		02 FB 000F4	CALLS	#2, SYS\$BINTIM	: 0297
		52		50 D0 000FB	MOVL	R0, STATUS	: 0303
		16		52 E9 000FE	BLBC	STATUS, 3\$	: 0305
				7E D4 00101	CLRL	-(SP)	: 0307
			0000V	CF 9F 00103	PUSHAB	TIMEOUT_AST	: 0308
			FF00	CD 9F 00107	PUSHAB	DELTA_TIME	: 0309
				7E D4 0010B	CLRL	-(SP)	: 0310
	00000000G	00		04 FB 0010D	CALLS	#4, SYS\$SETIMR	: 0311
		52		50 D0 00114	MOVL	R0, STATUS	: 0312
		03		52 E8 00117	BLBS	STATUS, 4\$	: 0313
			01EE	31 0011A	BRW	13\$	: 0314
10	00	6E		00 2C 0011D	MOVC5	#0, (SP), #0, #16, NFB	: 0315
			A0	AD 00122	MOVB	#23, NFB	: 0316
	A0	AD		17 90 00124	CLRQ	-(SP)	: 0317
				7E 7C 00128	CLRQ	-(SP)	: 0318
				7E 7C 0012A	CLRL	-(SP)	: 0319
				7E D4 0012C	PUSHAB	NFB_DESC	: 0320
			98	AD 9F 0012E	CLRQ	-(SP)	: 0321
				7E 7C 00131	PUSHAB	IOSB	: 0322
			90	AD 9F 00133	PUSHL	#56	: 0323
				38 DD 00136	MOVZWL	NET_CHANNEL, -(SP)	: 0324
		7E		6B 3C 00138	CLRL	-(SP)	: 0325
	00000000G	00		7E D4 0013B	CALLS	#12, SYS\$QIOW	: 0326
		56		0C FB 0013D	MOVL	R0, STATUS	: 0327
		07		50 D0 00144	BLBC	STATUS, 5\$	: 0328
		56	90	56 E9 00147	MOVZWL	IOSB, STATUS	: 0329
		17		AD 3C 0014A	BLBS	STATUS, 6\$	: 0330
		2C		56 E8 0014E	CMPL	STATUS, #44	: 0331
				56 D1 00151	BNEQ	7\$	: 0332
		7E		6E 12 00154	MOVZWL	NET_CHANNEL, -(SP)	: 0333
	00000000G	00		6B 3C 00156	CALLS	#1, SYS\$DASSGN	: 0334
		50	10000004	01 FB 00159	MOVL	#268435460, R0	: 0335
				8F D0 00160	RET		: 0336
				04 00167	MOVL	IOSB+4, IPID	: 0337
10	00	57	94	AD D0 00168	MOVC5	#0, (SP), #0, #16, NFB	: 0338
		6E		00 2C 0016C	MOVB	#34, NFB	: 0339
			A0	AD 00171	MOVB	#18, NFB+2	: 0340
	A0	AD		22 90 00173	MOVL	#302055440, NFB+4	: 0341
	A2	AD		12 90 00177	CLRB	NFB+3	: 0342
	A4	AD	12010010	8F D0 0017B	MOVC3	#16, P.AAJ, NFB+16	: 0343
			A3	AD 94 00183			: 0344
B0	AD	64	A8	10 28 00186			: 0345





69		01	FB	00255	CALLS	#1, LIB\$PUT_OUTPUT					
	3C	AE	9F	00258	PUSHAB	ASCII_NCB_DESC	0362				
	0120	C8	9F	0025B	PUSHAB	P.AAS					
6A		02	FB	0025F	CALLS	#2, FAO_BUFFER					
		50	DD	00262	PUSHL	R0					
69		01	FB	00264	CALLS	#1, LIB\$PUT_OUTPUT					
	34	AE	9F	00267	PUSHAB	FILESPEC	0363				
	0144	C8	9F	0026A	PUSHAB	P.AAU					
6A		02	FB	0026E	CALLS	#2, FAO_BUFFER					
		50	DD	00271	PUSHL	R0					
69		01	FB	00273	CALLS	#1, LIB\$PUT_OUTPUT					
	014C	C8	9F	00276	PUSHAB	P.AAW	0364				
6A		01	FB	0027A	CALLS	#1, FAO_BUFFER					
		50	DD	0027D	PUSHL	R0					
69		01	FB	0027F	CALLS	#1, LIB\$PUT_OUTPUT					
	0194	C8	9F	00282	PUSHAB	P.AAY	0365				
6A		01	FB	00286	CALLS	#1, FAO_BUFFER					
		50	DD	00289	PUSHL	R0					
69		01	FB	0028B	CALLS	#1, LIB\$PUT_OUTPUT					
	019C	C8	9F	0028E	PUSHAB	P.ABA	0366				
6A		01	FB	00292	CALLS	#1, FAO_BUFFER					
		50	DD	00295	PUSHL	R0					
69		01	FB	00297	CALLS	#1, LIB\$PUT_OUTPUT					
	2C	AE	9F	0029A	PUSHAB	PRCNAM	0369				
00000000G	00	01	FB	0029D	CALLS	#1, SYS\$SETPRN					
	52	50	DD	002A4	MOVL	R0, STATUS					
	51	52	E9	002A7	BLBC	STATUS, 13\$					
		AE	9F	002AA	PUSHAB	NCB_DESC	0373				
	44	C8	9F	002AD	PUSHAB	P.ABC					
00000000G	00	02	FB	002B1	CALLS	#2, LIB\$SET_LOGICAL					
	52	50	DD	002B8	MOVL	R0, STATUS					
	4D	52	E9	002BB	BLBC	STATUS, 13\$					
4F	AE	02	90	002BE	MOVB	#2, CMD_DESC+3	0375				
		AE	D4	002C2	CLRL	CMD_DESC+4	0376				
		AE	9F	002C5	PUSHAB	FILESPEC	0379				
		C8	9F	002C8	PUSHAB	P.ABE					
		AE	9F	002CC	PUSHAB	CMD_DESC					
00000000G	00	03	FB	002CF	CALLS	#3, STR\$CONCAT					
	52	50	DD	002D6	MOVL	R0, STATUS					
	2F	52	E9	002D9	BLBC	STATUS, 13\$					
38	BE	34	AE	01C0	C8	04	39	002DC	MATCHC	#4, P.ABG, FILESPEC, @FILESPEC+4	0383
						03	13	002E5	BEQL	10\$	
						04	DD	002E7	MOVL	#4, R3	
						04	C2	002EA	SUBL2	#4, R3	
						0C	13	002ED	BEQL	11\$	
						AE	9F	002EF	PUSHAB	FILESPEC	0386
00000000G	00	01	FB	002F2	CALLS	#1, LIB\$RUN_PROGRAM					
		0A	11	002F9	BRB	12\$					
		AE	9F	002FB	PUSHAB	CMD_DESC	0389				
00000000G	00	01	FB	002FE	CALLS	#1, LIB\$DO_COMMAND					
	52	50	DD	00305	MOVL	R0, STATUS					
	12	52	E8	00308	BLBS	STATUS, 14\$					
		52	DD	0030B	PUSHL	STATUS					
00000000G	00	01	FB	0030D	CALLS	#1, LIB\$SIGNAL					
50		52	10000000	8F	C9	00314	BISL3	#268435456, STATUS, R0			
				04	0031C	RET					
		01	DD	0031D	MOVL	#1, R0	0397				



16<sup>H</sup>-Sep-1984 01:39:23  
14-Sep-1984 12:49:31

Page 15  
(3)

RET

: 0399

[illegible]

```

: 403      0400 1 ROUTINE timeout_ast: NOVALUE =
: 404      0401 1
: 405      0402 1 --
: 406      0403 1
: 407      0404 1 This AST is called when our timer has expired. Since the
: 408      0405 1 DCLSERV QIO has not completed in the required amount of time,
: 409      0406 1 we assume that there are no more requests to be handled by this
: 410      0407 1 process, and we go away. This is done by cancelling the DECLSERV
: 411      0408 1 QIO.
: 412      0409 1
: 413      0410 1 Inputs:
: 414      0411 1
: 415      0412 1 net_channel = Network channel which has DECLSERV pending.
: 416      0413 1
: 417      0414 1 Outputs:
: 418      0415 1
: 419      0416 1 None
: 420      0417 1 ---
: 421      0418 1
: 422      0419 2 BEGIN
: 423      0420 2
: 424      0421 2 $CANCEL(CHAN = .net_channel); ! Cancel the DECLSERV QIO
: 425      0422 2
: 426      0423 1 END;

```

.EXTRN SYS\$CANCEL

```

                                0000 00000 TIMEOUT_AST:
                                .WORD Save nothing
                                MOVZWL NET_CHANNEL, -(SP)
                                CALLS #1, SYS$CANCEL
                                RET
00000000G 7E 0000' CF 3C 00002
00000000G 00 01 FB 00007
                                04 0000E

```

```

: 0400
: 0421
:
: 0423

```

; Routine Size: 15 bytes, Routine Base: \$CODE\$ + 0321



**! Issue read on mailbox**

0424  
0452  
0454

; Routine Size: 56 bytes, Routine Base: \$CODE\$ + 0330

NETWORK\_SERVER  
V04-000

K 6  
16-Sep-1984 01:39:23  
14-Sep-1984 12:49:31

VAX-11 BLISS-32 V4.0-742  
[NETACP.SRC]SERVER.B32;1

Page 18  
(5)[illegible]



```

: 460      0455 1 ROUTINE net_interrupt: NOVALUE =
: 461      0456 1
: 462      0457 1 ---
: 463      0458 1
: 464      0459 1 This AST routine is called when the outstanding QIO
: 465      0460 1 on the associated mailbox completes. If the interrupt
: 466      0461 1 indicates that the network is going down, then make us
: 467      0462 1 go away by canceling any I/O on the network channel
: 468      0463 1 (most likely a pending DECLSERV).
: 469      0464 1
: 470      0465 1 Inputs:
: 471      0466 1
: 472      0467 1     mbx_message = Mailbox message
: 473      0468 1     net_channel = Channel to network ACP
: 474      0469 1
: 475      0470 1 Outputs:
: 476      0471 1
: 477      0472 1     None
: 478      0473 1 ---
: 479      0474 1
: 480      0475 2 BEGIN
: 481      0476 2
: 482      0477 2 IF .mbx_message [0] EQL msg$_netshut ! If network shutting down,
: 483      0478 2 THEN
: 484      0479 3 BEGIN
: 485      0480 3     $DASSGN(CHAN = .net_channel); ! Cancel any pending DECLSERV I/O
: 486      0481 3     net_channel = 0; ! Mark channel no longer active
: 487      0482 3     RETURN; ! Do not re-issue mailbox read
: 488      0483 2     END;
: 489      0484 2
: 490      0485 2 issue_mailbox_read(); ! Issue another read on mailbox
: 491      0486 2
: 492      0487 1 END;

```

				0000 00000 NET_INTERRUPT:		
				.WORD	Save nothing	: 0455
	3B	0000'	CF 91 00002	CMPB	MBX_MESSAGE, #59	: 0477
			11 12 00007	BNEQ	1\$	
	7E	0000'	CF 3C 00009	MOVZWL	NET_CHANNEL, -(SP)	: 0480
00000000G	00		01 FB 0000E	CALLS	#1, SYSSDASSGN	
		0000'	CF B4 00015	CLRW	NET_CHANNEL	: 0481
			04 00019	RET		: 0479
AA AF		00	FB 0001A 1\$:	CALLS	#0, ISSUE_MAILBOX_READ	: 0485
			04 0001E	RET		: 0487

; Routine Size: 31 bytes, Routine Base: \$CODE\$ + 0368

```

: 494      0488 1 ROUTINE fao_buffer (ctrstr,args) =
: 495      0489 2 BEGIN
: 496      0490 2
: 497      0491 2 ---
: 498      0492 2
: 499      0493 2 This routine passes an ascii string through the FAO
: 500      0494 2 system service with any number of specified parameters.
: 501      0495 2
: 502      0496 2 ---
: 503      0497 2
: 504      0498 2 OWN
: 505      0499 2 desc : VECTOR[2] ; Result descriptor
: 506      0500 2 buf : VECTOR[512,BYTE]; ; Output buffer
: 507      0501 2
: 508      0502 2 MAP
: 509      0503 2 ctrstr : REF VECTOR[2],
: 510      0504 2 args : VECTOR[4];
: 511      0505 2
: 512      0506 2 desc[0] = 512; ; Set up result descriptor
: 513      0507 2 desc[1] = buf;
: 514      0508 2 $faol(ctrstr=.ctrstr,outlen=desc,outbuf=desc,prmlst=args);
: 515      0509 2 RETURN desc;
: 516      0510 1 END;

```

```

.PSECT $OWNS$,NOEXE,2
0008C DESC: .BLKB 8
00094 BUF: .BLKB 512
.EXTRN SYSS$FAOL
.PSECT $CODE$,NOWRT,2

```

```

0004 00000 FAO_BUFFER:
      52 0000' CF 9E 00002 .WORD Save R2 ; 0488
      62 0200 8F 3C 00007 MOVAB DESC, R2 ; 0506
04 A2 08 A2 9E 0000C MOVZWL #512, DESC ; 0507
      08 AC 9F 00011 MOVAB BUF, DESC+4 ; 0508
      52 DD 00014 PUSHAB ARGS
      52 DD 00016 PUSHL R2
      04 AC DD 00018 PUSHL R2
00000000G 00 04 FB 0001B PUSHL CTRSTR
      50 62 9E 00022 CALLS #4, SYSS$FAOL
      04 00025 MOVAB DESC, R0 ; 0509
RET ; 0510

```

; Routine Size: 38 bytes, Routine Base: \$CODE\$ + 0387



: 518 0511 1 END  
: 519 0512 0 ELUDOM

.EXTRN LIB\$SIGNAL

PSECT SUMMARY

Name	Bytes	Attributes
\$OWNS	660	NOVEC, WRT, RD, NOEXE, NOSHR, LCL, REL, CON, NOPIC, ALIGN(2)
\$SPLITS	452	NOVEC, NOWRT, RD, NOEXE, NOSHR, LCL, REL, CON, NOPIC, ALIGN(2)
\$CODES	941	NOVEC, NOWRT, RD, EXE, NOSHR, LCL, REL, CON, NOPIC, ALIGN(2)

Library Statistics

File	Total	Symbols Loaded	Percent	Pages Mapped	Processing Time
_\$255\$DUA28:[SYSLIB]STARLET.L32;1	9776	26	0	581	00:01.0
_\$255\$DUA28:[SHRLIB]NET.L32;1	1279	16	1	63	00:00.9

: Information: 1  
: Warnings: 0  
: Errors: 0

COMMAND QUALIFIERS

: BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/LIS=LISS:SERVER/OBJ=OBJ\$:SERVER MSRC\$:SERVER/UPDATE=(ENH\$:SERVER)

: Size: 941 code + 1112 data bytes  
: Run Time: 00:19.8  
: Elapsed Time: 00:38.3  
: Lines/CPU Min: 1554  
: Lexemes/CPU-Min: 22615  
: Memory Used: 252 pages  
: Compilation Complete



0279 AH-BT13A-SE  
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION  
CONFIDENTIAL AND PROPRIETARY